Design

Disk Scheduling Simulator

Sagalpreet Singh

7th November, 2021

Introduction

1. What's it?
2. Logic
3. Analysis
4. Results

What's it?

This program simulates the process of scheduling requests for disk access using Random, FIFO, SSTF, SCAN and CSCAN algorithms. Statistics like mean and standard deviation of response time for these algorithms are stored and analyzed and performance of these algorithms is discussed.

Program is run with different input parameters:

1. Rotational speed of disk, r revolutions per minute.

2. Average seek time, T in ms.

3. Sector size, N in bytes.

4. Disk Scheduling Algorithm (by its code).

These parameters are taken as command line arguments.

Parameters to gauge performance:

1. Minimum Response Time

2. Maximum Response Time

3. Average Response Time

4. Standard Deviation of Response Time

5. Throughput

Assumption: Disk Scheduling Algorithms themselves take no time.

Logic

- Address : Is defined by Platter, Sector and Track on the hard drive

- Request Generation : Requests are generated beforehand with the following attributes for each request: Spawn Time, Address, Number of Sectors. All the requests are uniformly generated with a constant time gap. This time gap is specified as a hyperparameter in the main.c file. The attributes for each request are randomly generated.

- The disk scheduling algorithms are implemented as follows:

    - A global time variable is maintained (initialized to 0)

    - It checks from all the pending requests (by comparing current time and spawn time of each request) to find the suitable one according to the algorithm. Processes it and increments the time accordingly.

    - Before the disk scheduler runs, it is checked if there is a pending request. If there is no such request, time is advanced to move to the nearest next time when a request spawns.

- The number of requests is also a hyperparameter set in main.c

- The hard disk stores the current location of the head (which sector and track it lies on). Only one such instance is maintained as it will be the same for all the platters (this is how most hard disks work nowadays). In addition to this, the direction in which the head is currently moving is also stored.

- Proper care has been taken to rotate the disk at the time when the head is moving. (Disks keep on rotating and don't stop unless powered off). Thus it may happen that although the disk was in the right position but head was not on proper track and in time when head came to proper track, the disk had rotated from its original direction, so more time will be taken now.

- Two things are stored in the results directory:

    - data.txt : It contains the statistics

    - requests.txt : It contains the request queue that was generated, one in each line (without word wrap of course)

    * The data.txt file may not be nice formatted if run directly without bash script as instructed in further section.

- Default Hyperparameters
    - SURFACES = 4
    - CYLINDERS = 25
    - SECTORS = 20
    - NUM_REQUESTS = 1000
    - TIME_GAP = 5

* ms is used as unit of time everywhere except throughput for which it is "number of requests per second"

Analysis

1) FIFO algorithm seems to be the best fit (at least for the hyperparameters selected) because it has the lowest value of mean response time which means throughput is high. Also the standard deviation is the lowest which implies that there is least starvation of requests in this case.
2) Random is not a good choice because it has performed very poorly in all the scenarios.
3) SSTF performs well but its performance varies largely with rotation speed of the disk. This is expected because of the cases in which even though the head might have to move less but the disk had to make a lot of turns. With high rotation speeds, SSTF may outperform FIFO as the difference is already very less at 15000 RPM.
4) SCAN and CSCAN algorithm's behavior are very similar to that of SSTF but CSCAN is worse than either of those. SCAN tends to outperform SSTF.

Keeping all of the above in mind (for upto 15000 RPM):

FIFO > SCAN > SSTF > CSCAN > Random

For higher RPM's, the order might look like:

SCAN > SSTF > CSCAN > FIFO > Random

Results

The table can be viewed in data.txt in the results directory.